EECE.3220: Data Structures

Spring 2017

Syllabus

Course Meetings

Section 202: MWF 1-1:50, Ball 412

Course Website

Main page: http://mjgeiger.github.io/eece3220/sp17/

Schedule: http://mjgeiger.github.io/eece3220/sp17/schedule.htm

Course Discussion Group

All course announcements will be posted on the discussion group—you are responsible for checking the board regularly or enabling direct e-mail updates from Piazza.

Sign up link: http://piazza.com/uml/spring2017/eece3220

Instructor

Dr. Michael Geiger
E-mail: Michael_Geiger@uml.edu
Office: Perry Hall 118A
Phone: 978-934-3618 (x43618 on campus)
Office hours: Monday 9:30-11, Wednesday 9:30-11, Thursday 1:30-3

During the above hours, student questions are my top priority. I will also be in my office MWF 11-11:45, and I am available by appointment at other times.

Feel free to stop by my office, e-mail me questions, or schedule a one-on-one appointment. Office hours are subject to change.

Textbook

Larry Nyhoff, *ADTs, Data Structures, and Problem Solving with C++*, 2nd edition, 2005, Pearson/Prentice Hall.
ISBN: 0-13-140909-3

Course Overview

Catalog Description: Covers algorithms and their performance analysis, data structures, abstraction, and encapsulation. Introduces structures and their physical storage representation. Studies stacks, queues, linked lists, trees, graphs, heaps, priority queues, and hashing. Discusses efficient sorting (quicksort and heapsort) and introduces experimental analysis of algorithms as applied to engineering applications. Examines several design issues, including selection of structures based on what operations need to be optimized (insertion, deletion, traversal, searching, sorting, evaluation), encapsulation of algorithms using class and template techniques, and how and when to use recursion (versus explicit stack-based techniques). Laboratories include programming of data structures in C++ and Java applied to Engineering.

Prerequisites: EECE.2160: ECE Application Programming

Course Objectives: By the end of this course, you should understand and be able to use all of the following:

1. **C++ Programming:** Fundamentals of the C++ programming language
2. **Data Structures:** Common data structures, including arrays, vectors, stacks, queues, linked lists, trees, and hash tables
3. **Algorithmic Complexity:** Analyzing the performance of data structures and algorithms
4. **Object-Oriented Programming:** Classes, objects, templates, inheritance

Grading: Grades will be computed on an A to F scale; no A+ grades will be assigned, in accordance with UMass Lowell policy. The weights assigned to the various items are:

Programming/homework assignments	55%
Exam 1	15%
Exam 2	15%
Exam 3	15%

Incomplete grades will only be given in exceptional situations, and the student must be passing the class at the time the grade is requested.

The following rubric describes how grades will be assigned if no grading curve is applied. A grading curve may be used at the instructor's discretion, depending on the overall course average at the end of the term. Grades will not be curved down, meaning that the table below describes the minimum letter grade you will earn for a final average in each of the ranges shown:

Range	**Grade**	**Range**	**Grade**
> 92	A	78-79	C+
90-92	A-	73-77	C
88-89	B+	70-72	C-
83-87	B	68-69	D+
80-82	B-	60-67	D
		< 60	F

Programming/homework assignments: Typically, you will have about one week to complete each assignment. All assignments will be graded according to the program grading guidelines, to be distributed separately. Late assignments will lose 2^{n-1} points per day, including weekends and holidays. You will submit your work by uploading your files directly to your Dropbox folder.

For each assignment, you will be allowed one resubmission to improve your grade without penalty. You must resubmit your code by the given deadline for that assignment; late penalties will apply to late resubmissions. Note that the resubmission policy does not allow you to avoid penalties when the original submission is late (e.g., an assignment losing 4 points for a late initial submission has a maximum possible score of 96 for the resubmission). See the grading guidelines for more details.

Exams: Make-up exams will only be offered in exceptional circumstances. You must notify your instructor as early as possible in order to determine an appropriate make-up date.

Class participation: You are responsible for all material discussed or announced in class. You are expected to attend class regularly and participate in any in-class discussions, as such exercises are essential to your learning. Although lecture attendance is not explicitly required, regular attendance will improve your understanding of the course concepts.

Academic Honesty

All assignments and exams must be completed individually unless otherwise specified. You may discuss concepts or material covered in class, but may not share any details of your solutions to assigned problems, including algorithms and code. Plagiarism (in this course, copying code from an outside source) is also unacceptable and will be treated as an instance of cheating.

Students are allowed to discuss assignments in general terms and to help one another fix specific errors—examples include compiler errors or output formatting. In this case, students are required to note that they received assistance from a classmate by listing that person's name and the nature of their assistance as part of their assignment header. However, any sharing of code—even when used strictly to help a classmate solve a specific error—is a violation of the academic honesty policy.

Any assignment or portion of an assignment that violates this policy will receive a grade of zero for all parties concerned. Depending on the severity of the infraction, or in cases of repeat violations, additional penalties may be given at the instructor's discretion, up to and including a failing grade in the course.

Further information on the university Academic Integrity policy can be found at:

http://www.uml.edu/Catalog/Undergraduate/Policies/Academic-Policies/Academic-Integrity.aspx

Course Schedule

This schedule contains a tentative schedule of topics we will cover throughout the term; the course website will contain the most up-to-date version. The web page will also describe which section(s) of the textbook are associated with each lecture and the due dates for each assignment.

Please note that several days are denoted as "PE#"—in these classes, we will do an in-class programming exercise. While students will be able to participate even if they do not have a computer, I suggest anyone with a laptop bring it to class on these days.

Please note that the exam dates are fixed—the first exam will be held on **Friday, February 17 in class**, the second exam will be held on **Friday, March 31 in class**, and the third exam will be held **during finals, at a date/time to be determined**.

Week	Date (M)	Lecture Topics
1	1/16	*No Monday lecture—Martin Luther King, Jr. Day* 1. Course introduction; going from C to C++ 2. C to C++: I/O, structs
2	1/23	3. C to C++: functions, arg. passing, declarations 4. Reviewing separate compilation 5. Comments; I/O manipulators
3	1/30	6. Algorithmic complexity 7. Algorithmic complexity (cont.); abstract data types 8. ADTs (cont.); overloaded functions & default arguments
4	2/6	9. Classes: members, accessor and mutator functions 10. Classes: constructors, I/O member functions 11. Calling objects, inline member functions, assert()
5	2/13	12. Array-based lists 13. Exam 1 Preview **Friday, 2/17: EXAM 1**
6	2/20	*No Monday lecture—Presidents Day* 14. Dynamic memory allocation in C++ *(Tuesday, 2/21)* 15. Linked lists 16. Circular linked lists
7	2/27	17. Doubly linked lists; const in C++ 18. Stacks 19. Stacks (continued)
8	3/6	20. Queues 21. Queues (continued) 22. Recursion
9	3/13	*No classes—Spring Break*

Course Schedule (continued)

Week	Date (M)	Lecture Topics
10	3/20	23. Recursive algorithms and complexity 24. Function/class templates; binary trees 25. Binary trees (continued)
11	3/27	26. File I/O; strings 27. Exam 2 Preview **Friday, 3/31: EXAM 2**
12	4/3	28. Simple sorting algorithms 29. Heaps *Wednesday, 4/5: Last day to withdraw* 30. Heaps and priority queues
13	4/10	31. Priority queues; more sorting algorithms 32. Hash tables 33. Hash tables (continued)
14	4/17	*No Monday lecture—Patriots' Day* 34. Composition and inheritance 35. Inheritance (continued)
15	4/24	36. C++ exceptions 37. Topics TBD 38. Exam 3 Preview *Classes end Friday, 4/28*
	TBD	**EXAM 3 (DATE/TIME TBD)**